Question 1 (a): Implement KNN Classifier

1. For k = 1, what is the predicted label for the following example in the test set: It leaves little doubt that Kidman has become one of our best actors . (This is line 18 of the test file.)

   Predicted label for the following example in the test set is: 1

2. What is the confusion matrix (on the test set) for k = 1?

r/y		Predicted (y)	
		1	0
Correct (r)	1	True Positive = 209	False Negative = 64
	0	False Positive = 134	True Negative = 93

3. Report the accuracy, the true positive rate, and the false positive rate, on the test set for k = 1.

   Accuracy = 60.4
   True Positive Rate = 0.7655677655677655
   False Positive Rate = 0.5903083700440529

4. For k = 5, what is the predicted label for the following example in the test set: It leaves little doubt that Kidman has become one of our best actors . (This is line 18 of the test file.)

   Predicted label for the following example in the test set is: 1

5. What is the confusion matrix (on the test set) for k = 5?

r/y		Predicted (y)	
		1	0
Correct (r)	1	True Positive = 212	False Negative = 61
	0	False Positive = 136	True Negative = 91

6. Report the accuracy, the true positive rate, and the false positive rate, on the test set for k = 5.

   Accuracy = 60.6
   True Positive Rate = 0.7765567765567766
   False Positive Rate = 0.5991189427312775

7. What is the accuracy on the test set for k = 5?

   Accuracy = 60.6

8. Suppose we used the very simple Zero-R classifier on this dataset, rather than k-NN. That is, we classify all examples in the test set as belonging to the class that is more common in the training set. What is the resulting confusion matrix (on the test set)?

r/y		Predicted (y)	
		1	0
Correct (r)	1	True Positive = 273	False Negative = 0
	0	False Positive = 227	True Negative = 0

Question 1 (b):
In the dataset we are using here, the examples (comments) are all fairly short, containing relatively few tokens. Suppose that we had a dataset consisting of documents of very different lengths, ranging from e.g., 10 tokens in length, to 10,000 tokens in length. If we applied k-NN to such a dataset, the distance function we are using here might not be a good choice. Why not?

If the token length is large, the distance function we are using here might not be a good choice because of below reasoning:

- The Calculations and Comparisons will increase, hence reducing the efficiency of KNN Classifier.
- There will more redundant comparisons of short words, stop words as they are used often in a text. E.g. "a", "the", "his", "her", "he", "she", "all". This will affect the distance largely and further, will affect KNN Classifier's accuracy.
- The Run time of Classifier will increase drastically as the K value increases.

Question 1 (c): Cross Validation

1. For each of the 3 values of k, what is the cross-validation accuracy?

   For K = 3
   Accuracy = 66.06666666666666

   For K = 7
   Accuracy = 65.86666666666666

   For K = 99
   Accuracy = 61.199999999999996

2. Take the k that had the highest cross-validation accuracy. Run k-NN on the entire training set for this value of k, and then test on the test set. Give the confusion matrix and the accuracy (for the test set).

   Highest K = 3

r/y		Predicted (y)	
		1	0
Correct (r)	1	True Positive = 212	False Negative = 61
	0	False Positive = 144	True Negative = 83

   Accuracy = 59.0

Question 1(d):

1. Describe your distance function. How is the distance between two comments computed? Include an example in your explanation.

We have used two distance functions where I have achieved higher accuracy in the model than our default distance function.

1. Refined Intersection Distance

This distance function manipulates the string text by removing redundant texts which can be considered for comparison and then calculates the distance using our same formula. For this method, we will get different distance scores for some of the text which refines our accuracy. The following function performs below manipulation before taking distance:

- **Remove special characters, numbers, punctuations:** We can also think of getting rid of the punctuations, numbers and even special characters since they wouldn't help in differentiating different kinds of reviews. Hence, it is better to remove them from the text.
- **Remove Short Words:** Most of the smaller words do not add much value. For example, 'pdx', 'his', 'all', 'the', 'her'. So, we will try to remove them as well from our data. We were little careful here in selecting the length of the words which we want to remove. So, we decided to remove all the words having length 3 or less. For example, terms like "hmm", "oh" are of very little use. It is better to get rid of them.
- **Stemming:** Stemming is a rule-based process of stripping the suffixes ("ing", "ly", "es", "s" etc) from a word. For example, for example – "play", "player", "played", "plays" and "playing" are the different variations of the word – "play".
- **Removing of Stop Words:** Removing stop words like 'a', 'the' will help to calculate similarity distances more effectively.

2. Jaccard Similarity Distance¶

The Jaccard similarity index (sometimes called the Jaccard similarity coefficient) compares members for two sets to see which members are shared and which are distinct. It's a measure of similarity for the two sets of data, with a range from 0% to 100%. The higher the percentage, the more similar the two populations.

The formula to find the Index is:

```
Jaccard Index = (the number in both sets)/(the number in either set)*100
```

A simple example using set notation: How similar are these two sets?

```
A = {0,1,2,5,6}

B = {0,2,3,4,5,7,9}

J(A,B) = |A∩B| / |A∪B| = |{0,2,5}| / |{0,1,2,3,4,5,6,7,9}| = 3/9 = 0.33.
```

We have used Jaccard distance which is a measure of how dissimilar two sets are. It is the complement of the Jaccard index and can be found by subtracting the Jaccard Index from 100%.

```
For the above example, the Jaccard distance is 1 - 33.33% = 66.67%.
```

In set notation, subtract from 1 for the Jaccard Distance:

```
D(X,Y) = 1 - J(X,Y)
```

PTO

2. Why did you think that your distance function would do better than the first one?

   In the first distance function, we consider complete set of tokens from a text and then calculate distance. In this way, we also consider redundant tokens which are not necessary for comparison of text for sentimental analysis.

   In my distance calculation methods, especially "Refined Intersection Distance", we have removed the redundant texts which gives us only relevant tokens for comparing and calculating distance.

   Also, in the first distance function, we are not considering weights/length/size of two strings.

   Jaccard Similarity distance is weighing over intersection as well union of two strings. An alternate interpretation of the Jaccard distance is as the ratio of the size of the symmetric difference. Therefore, weights are calculated in symmetric fashion and can be correctly classified by KNN as it will have more distributed values.

3. What is the confusion matrix for k = 1?
   Calculations done under **"Running for all Distances for Comparison"** Section

   Using Refined Intersection Distance:

r/y		Predicted (y)	
		1	0
Correct (r)	1	True Positive = 229	False Negative = 44
	0	False Positive = 92	True Negative = 135

   Using Jaccard Distance:

r/y		Predicted (y)	
		1	0
Correct (r)	1	True Positive = 206	False Negative = 67
	0	False Positive = 112	True Negative = 115

4. Report the accuracy, the true positive rate, and the false positive rate, on the test set for k = 1?
   (Calculations done under "Running for all Distances for Comparison" Section)

   Using Refined Intersection Distance:

   Accuracy = 72.8
   True Positive Rate = 0.8388278388278388
   False Positive Rate = 0.4052863436123348

   Using Jaccard Distance:

   Accuracy = 64.2
   True Positive Rate = 0.7545787545787546
   False Positive Rate = 0.4933920704845815

PTO

5. What is the confusion matrix for k = 5?

   Calculations done under **"Running for all Distances for Comparison"** Section

   Using Refined Intersection Distance:

r/y		Predicted (y)	
		1	0
Correct (r)	1	True Positive = 236	False Negative = 37
	0	False Positive = 93	True Negative = 134

   Using Jaccard Distance:

r/y		Predicted (y)	
		1	0
Correct (r)	1	True Positive = 220	False Negative = 53
	0	False Positive = 113	True Negative = 114

6. Report the accuracy, the true positive rate, and the false positive rate, on the test set for k = 5?
   Calculations done under **"Running for all Distances for Comparison"** Section

   Using Refined Intersection Distance:

   Accuracy = 74.0
   True Positive Rate = 0.8644688644688645
   False Positive Rate = 0.40969162995594716

   Using Jaccard Distance:

   Accuracy = 66.8
   True Positive Rate = 0.8058608058608059
   False Positive Rate = 0.4977973568281938

7. Did your distance function achieve higher accuracy (for k = 1 and k = 5) than the first distance function?
   For the Comparison shown in "Running for all Distances for Comparison" Section, both of our distance functions achieve higher accuracy than first distance function:

   Using Default Distance:

   - For K = 1, Accuracy = 60.4
   - For K = 5, Accuracy = 60.6

   Using Refined Intersection Distance:

   - For K = 1, Accuracy = 72.8
   - For K = 5, Accuracy = 74.0

   Using Jaccard Distance:

   - For K = 1, Accuracy = 64.2
   - For K = 5, Accuracy = 66.8